ICC - *APONTADORES E ARQUIVOS*

PROF. FERNANDO W CRUZ

Roteiro da aula

- ALOCAÇÃO DINÂMICA DE MEMÓRIA
- PASSANDO PARÂMETROS PARA PROGRAMAS
- ARQUIVOS (INTRODUÇÃO)

Alocação dinâmica de memória

1. Confira o programa abaixo e verifique se há algum erro significativo:

```
#include <stdio.h>
#include <stdlib.h>

main () {

  int *nota;

  printf("Digite sua nota : ");
  scanf("%d", nota);
  printf("A nota digitada foi ... : %d\n", *nota);

} /* fim-main */
```

- Essa declaração não aloca um endereço para o número inteiro.
- Apenas declara que **nota** deve apontar para um endereço que conterá um número inteiro.

Provavelmente o programa acusará erro, pois não houve alocação de memória!

Alocação dinâmica de memória

Versão correta... (com alocação dinâmica de memória): função ***malloc();***

```c
#include <stdio.h>
#include <stdlib.h>

main () {

  int *nota;

  nota = (int *) malloc ( sizeof(int));

  printf("Digite sua nota : ");
  scanf("%d", nota);
  printf("A nota digitada foi ... : %d\n", *nota);

} /* fim-main */
```

Esse comando faz alocação dinâmica de memória para comportar um inteiro. Esse endereço de memória será apontado pela variável nota.

O programa deve conter uma função para alocar espaço de memória para comportar o inteiro que vai ser lido.

Alocação dinâmica de memória: uso de *malloc*

2. Dado o exercício do *slide* seguinte, altere-o para que contenha alocação dinâmica de memória (retomando o exercício da aula anterior), ou seja:

Modifique a declaração:

```
main () {
  struct formulario aluno[TAM_TURMA];
  int i, j;
```

para:

```
main () {
  struct formulario *aluno[TAM_TURMA];
  int i, j;
```

Observe que o vetor aluno passa a ser um vetor cujo conteúdo refere-se a endereços de memória que deverão comportar o tipo ***struct formulario***.

Observe que as funções principal (*main*) e leia_registro precisarão sofrer alterações para comportar a mudança proposta.

Passagem de *structs* por referência

Versão 9:

```
void leia_registro (struct formulario *s) {
   printf("Matricula : ");
   leia_string(s->matricula);
   printf("Nome  : ");
   leia_string(s->nome);
   printf("Endereço  : ");
   leia_string(s->endereco);
   printf("Idade     : ");
   scanf("%d", &s->idade);
} /* fim-leia_string */

main () {
   struct formulario aluno[TAM_TURMA];
   int i, j;

   i = 0;
   while (i<TAM_TURMA) {
        leia_registro(&aluno[i]);
        i++;
   } /* fim-while */
   printf("\n\n-------------------------------------------------------\n");
   printf("UnB - Universidade de Brasilia / Campus FGA-Gama\n");
   printf("Disciplina: ICC - Introdução à Ciência da Computação, turma BB\n");
   printf("-------------------------------------------------------\n");
   printf("Matrícula\tNome\t\t\tEndereço\t\t\t\tIdade\n");
   printf("-------------------------------------------------------\n");
   i = 0;
   while (i<TAM_TURMA) {
        printf("%s\t",aluno[i].matricula);
        printf("%s",aluno[i].nome);
        for (j=0; j < (TAM_NOME - strlen(aluno[i].nome)); j++) {
            printf("%c", ' ');
        }
        printf("%s",aluno[i].endereco);
        for (j=0; j < (TAM_END - strlen(aluno[i].endereco)); j++) {
            printf("%c", ' ');
        }
        printf("%d\n",aluno[i].idade);
        i++;
   } /* fim-while */
   printf("-------------------------------------------------------\n");
} /* fim-main */
```

```
void leia_string(char *s) {
   int c, i;
   char *p;
   c = getchar();
   if (c =='\n') {
        c =getchar();
   } /* fim-if */
   i = 0;
   p=s;
   while (c!='\n') {
        (*s) = c;
        c = getchar();
        i++;
        s=p+i;
   } /* fim-while */
   (*s) = '\0';
} /* fim-leia_string */
```

Resposta possível:

Versão 10:

Alocação na função chamada (menos sugerido)

```
struct formulario * leia_registro () {
   struct formulario *s;
   s = (struct formulario *) malloc (sizeof (struct formulario));
   printf("\nMatricula : ");
   leia_string(s->matricula);
   printf("Nome  : ");
   leia_string(s->nome);
   printf("Endereço  : ");
   leia_string(s->endereco);
   printf("Idade     : ");
   scanf("%d", &s->idade);
   return(s);
} /* fim-leia_registro */
main () {
   struct formulario *aluno[TAM_TURMA];
   int i, j;

   i = 0;
   while (i<TAM_TURMA) {
        aluno[i]= leia_registro();
        i++;
   } /* fim-while */
   printf("\n\n-------------------------------------------------\n");
   printf("UnB - Universidade de Brasilia / Campus FGA-Gama\n");
   printf("Disciplina: ICC - Introdução à Ciência da Computação, turma BB\n");
   printf("-------------------------------------------------\n");
   printf("Matrícula\tNome\t\t\tEndereço\t\t\t\tIdade\n");
   printf("-------------------------------------------------\n");
   i = 0;
   while (i<TAM_TURMA) {
        printf("%s\t",aluno[i]->matricula);
        printf("%s",aluno[i]->nome);
        for (j=0; j < (TAM_NOME - strlen(aluno[i]->nome)); j++) {
            printf("%c", ' '); }
        printf("%s",aluno[i]->endereco);
        for (j=0; j < (TAM_END - strlen(aluno[i]->endereco)); j++) {
            printf("%c", ' '); }
        printf("%d\n",aluno[i]->idade);
        i++;
   } /* fim-while */
   printf("-------------------------------------------------\n");
} /* fim-main */
```

Aqui, leia_registro passou a ser uma função com retorno de um endereço onde estão os dados preenchidos.

Outra resposta possível

Versão 11:

Alocação na função principal (mais sugerido)

```
void leia_registro (struct formulario *s) {

   printf("\nMatricula : ");
   leia_string(s->matricula);
   printf("Nome  : ");
   leia_string(s->nome);
   printf("Endereço  : ");
   leia_string(s->endereco);
   printf("Idade     : ");
   scanf("%d", &s->idade);
} /* fim-leia_registro */
```

Aqui, **leia_registro** é uma *procedure* que preenche o endereço apontado pelo apontador **s** (ou aluno[i]).

```
main () {
   struct formulario *aluno[TAM_TURMA];
   int i, j;

   i = 0;
   while (i<TAM_TURMA) {
        aluno[i] = (struct formulario *) malloc (sizeof ( struct formulario ));
        leia_registro(aluno[i]);
        i++;
   } /* fim-while */
   printf("\n\n-------------------------------------------------------\n");
   printf("UnB - Universidade de Brasilia / Campus FGA-Gama\n");
   printf("Disciplina: ICC - Introdução à Ciência da Computação, turma BB\n");
   printf("-------------------------------------------------------\n");
   printf("Matrícula\tNome\t\t\tEndereço\t\t\t\tIdade\n");
   printf("-------------------------------------------------------\n");
   i = 0;
   while (i<TAM_TURMA) {
        printf("%s\t",aluno[i]->matricula);
        printf("%s",aluno[i]->nome);
        for (j=0; j < (TAM_NOME - strlen(aluno[i]->nome)); j++) {
            printf("%c", ' '); }
        printf("%s",aluno[i]->endereco);
        for (j=0; j < (TAM_END - strlen(aluno[i]->endereco)); j++) {
            printf("%c", ' '); }
        printf("%d\n",aluno[i]->idade);
        i++;
   } /* fim-while */
   printf("-------------------------------------------------------\n");
} /* fim-main */
```

Passando parâmetros para um programa

Exercício: Digite o programa abaixo e nomeie o arquivo fonte como **prog_fonte.c**

```
#include <stdio.h>
#include <stdlib.h>

main( int argc, char *argv[]) {
  int       i;

  printf("argc     = %d\n", argc);
  for (i=0; i<argc; i++) {
    printf("argv[%d] = %s\n", i, argv[i]);
  } /* fim-for */

} /* fim programa */
```

- **argc** = contém o número de argumentos do programa, incluindo o próprio nome do programa.
- **argv** = é um *array* de *strings*; em cada posição é passado um argumento; na primeira posição (índice 0) é sempre passado o nome do programa.

Em seguida, execute os passos abaixo:

a) Compile o programa da seguinte forma: ***gcc prog_fonte.c -o prog_objeto***

b) Execute o programa da seguinte forma: ***prog_objeto alo mundo***

Descubra quais os valores de **argc** e **argv** para essa execução.

Manipulação de arquivos (I)

Os *streams* mais comuns são os que ficam associados a arquivos armazenados em disco. A primeira operação necessária para trabalhar com esse tipo de streams é uma operação de abertura, efectuada com a função fopen():

```
FILE *fopen(char *name, char *mode);
```

A função fopen() retorna um apontador para uma estrutura FILE. O parâmetro name é o nome do arquivo armazenado no disco que se pretende acessar. O parâmetro mode controla o tipo de acesso. Se o arquivo especificado não puder ser acessado, por qualquer razão, a função retornará um apontador nulo (NULL). Os modos (parâmetro mode) principais incluem:

"r" - apenas leitura;
"w" - apenas escrita, cria sempre um novo arquivo mesmo que já exista;
"a" - permite acrescentar nova informação a arquivos já existentes.

É possível acrescentar aos designadores de modo as letras 't' ou 'b', que especificam o tipo de informação do arquivo: textual ou binária, respectivamente. O apontador retornado pela função deve ser guardado, uma vez que é necessário como parâmetro para todas as funções de acesso ao stream assim aberto.

Manipulação de arquivos (II)

Algumas opções de abertura de arquivos:

Modo	**Efeito**
“r”	Abre arquivo texto para leitura.
“w”	Abre arquivo texto para escrita. Se estiver presente, será sobreposto.
“a”	Abre um arquivo existente para inserção de novos dados.
“r+”	Abre arquivo texto para leitura e gravação. O arquivo deve existir e pode ser atualizado.
“w+”	Abre um arquivo texto para leitura e gravação. Se o arquivo existir, será sobreposto. Se não existir, será criado.
“a+”	Abre arquivo texto para atualizações e para adicionar dados ao fim do arquivo existente ou um novo arquivo será criado.

Manipulação de arquivos (III)

Declarando um arquivo e escrevendo informações, caractere a caractere (função ***putc***)

Comando de gravação de caracteres no arquivo apontado por **fd**, onde:

- **c** = caractere digitado: colhido pelo *getchar()*;
- **fd** = apontador para o FILE arq2.txt

```c
#include <stdio.h>
#include <stdlib.h>

main() {

  FILE * fd;
  char c;

  fd = fopen("arq2.txt", "w");
  c = getchar();
  while (c!='\n' ) {
      putc(c, fd);
      c = getchar();
  } /* fim-while */
  fclose(fd);
} /* fim-main */
```

Ao final da execução vai ser criado um arquivo com o nome arq2.txt contendo os caracteres digitados.

Manipulação de arquivos (IV)

Declarando um arquivo e lendo informações, caractere a caractere (função ***getc***)

Comando de leitura de caracteres do arquivo apontado por **fd**, onde:

- **c** = caractere lido do arquivo
- **fd** = apontador para o FILE arq2.txt
- **EOF** = Indicador de fim de arquivo (definido em <*stdio.h*>

```
#include <stdio.h>
#include <stdlib.h>

main() {

   FILE * fd;
   char c;

   fd = fopen("arq2.txt", "r");
   system("clear");
   c = getc(fd);
   while ( c != EOF ) {
       printf("%c", c);
       c = getc(fd);
   } /* fim-while */
   fclose(fd);
} /* fim-main */
```

Esse programa abre o arquivo com o nome arq2.txt e apresenta os caracteres desse arquivo na tela do computador.

Manipulação de arquivos (IV)

Declarando um arquivo e gravando informações, linha a linha (função ***fputs***)

Comando de gravação de linhas (lidas com gets à partir do teclado) no arquivo apontado por **fd**, onde:

- **linha** = *String* a ser gravada no arquivo
- **fd** = apontador para o FILE arq3.txt

Obs.: O caracter '\n' promove salto de linha.

```c
#include <stdio.h>
#include <string.h>

main() {

   FILE * fd;
   char linha[81];
   int tam_linha;

   fd = fopen("arq3.txt", "w");
   gets(linha);
   tam_linha = strlen(linha);
   while ( tam_linha > 0 ) {
        fputs(linha, fd);
        fputs("\n", fd);
        gets(linha);
        tam_linha = strlen(linha);
   } /* fim-while */
   fclose(fd);
} /* fim-main */
```

Ao final da execução vai ser criado um arquivo com o nome arq3.txt contendo as linhas digitadas no teclado (o programa para quando é digitado um <enter> no inicio da linha).

Manipulação de arquivos (V)

Declarando um arquivo e lendo informações, linha a linha (função ***fgets***)

Comando de leitura de linhas do arquivo apontado por **fd**, onde:

- **linha** = *String* com informações lidas do arquivo (limite de 80 caracteres)
- **fd** = apontador para o arquivo (FILE) arq3.txt
- suc_leitura = retorno da função fgets indicando sucesso ou não de leitura.

```c
#include <stdio.h>
#include <string.h>

main() {

   FILE * fd;
   char linha[81];
   char * suc_leitura;

   fd = fopen("arq3.txt", "r");
   suc_leitura = fgets(linha, 80, fd);
   while ( suc_leitura != NULL ) {
       printf("%s", linha);
       suc_leitura = fgets(linha, 80, fd);
   } /* fim-while */
   fclose(fd);
} /* fim-main */
```

Esse programa abre o arquivo com o nome arq3.txt e apresenta as linhas desse arquivo na tela do computador.